# A BRANCH-AND-BOUND ALGORITHM FOR THE SOLUTION OF SEQUENCE DEPENDENT ROUTING PROBLEMS

Michael Joseph DeHaemer

# United States
# Naval Postgraduate School


# THESIS

A BRANCH-AND-BOUND ALGORITHM FOR THE SOLUTION
OF
SEQUENCE DEPENDENT ROUTING PROBLEMS

by

Michael Joseph DeHaemer

April 1970

# A Branch-and-Bound Algorithm for the Solution of Sequence Dependent Routing Problems

by

Michael Joseph DeHaemer
Lieutenant Commander, United States Navy
B.S., University of Notre Dame, 1960

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
April 1970

ABSTRACT

A branch-and-bound algorithm, which finds the optimal route through n nodes when a different cost matrix occurs after each arc in the sequence is traversed, is presented. The route may begin at any node and must pass through each of the n nodes exactly once. The objective is to minimize total cost in traversing (n-1) arcs of the route. The cost of traversing each arc is $r_{ij}^{k}$, which is a function of the distance between nodes i and j and a function of the $k^{th}$ position in the sequence of arcs forming the route.

The algorithm is presented in step-by-step detail and illustrated by flow chart and examples. A modification for symmetric $(r_{ij}^{k})$ improves the efficiency of the algorithm.

A trade-off between computation time and storage requirements may be accomplished by alternate branching policies. Suboptimal solutions may be obtained with reduced computation.

TABLE OF CONTENTS

# I. INTRODUCTION

The algorithm presented in this thesis uses the branch-and-bound technique to find the optimal route through n nodes, which passes through each node exactly once, while any two of the nodes may be chosen for the start and finish of the route. The objective is to minimize total cost in traversing the (n-1) arcs of the route. The cost of traversing each arc if $r_{ij}^{k}$, which is a function of the distance between nodes i and j and a function of the $k^{th}$ position in the sequence of arcs forming the route.

The routing problem that is described and solved is closely related to the traveling salesman problem. It differs in that the route is not closed and that a different cost matrix occurs after each arc in the sequence is traversed.

Since the branch-and-bound methodology is used successfully by Little, et al, [Ref. 1] for solving the traveling salesman problem and is recommended by Bellmore and Nemhauser [Ref. 2] for medium-sized traveling salesman problems of 13 to 70 nodes, it is expected that the routing algorithm which is based on branch-and-bound concepts will provide an efficient solution to the sequence dependent routing problem and will serve as an alternative to dynamic programming with its large computer storage requirements.

As will be seen, the algorithm successfully solves any of the general class of sequence dependent routing problems for which the

matrices of all possible costs for traversing arcs can be provided at each stage in the sequence of arcs which form a complete route.

## II. THE BRANCH-AND-BOUND METHOD

### A. GENERAL CHARACTERISTICS

Virtually all programming techniques, including linear programming, dynamic programming, backtrack programming [Ref. 3], and branch-and-bound, contrive to find a vector, $\overline{X}^* = (x_1^*, x_2^*, \ldots, x_n^*)$, which optimizes an objective function $f(\overline{X})$. By some search rule elements are selected from the product space $X_1 x X_2 x \ldots x X_n$ of n "selection spaces" with $x_i$ an element of $X_i$. Hopefully the searching procedure will be considerably more efficient than the trial of all possible vectors in the product space.

The branch-and-bound search procedure consists of selectively partitioning the product space and then computing bounds on the objective function for each subset of the partition without enumerating all vectors in the subset. For the case in which $f(\overline{X})$ is to be minimized, the branch-and-bound technique first constructs a lower bound on $f(\overline{X})$ for all possible vectors in the product space. In the next step of the branch-and-bound procedure, the set of all possible solution vectors is partitioned and a tightened lower bound on $f(\overline{X})$ for each subspace of the product space is calculated.

As an example of these two steps, consider the problem of minimizing $f(x_1, x_2, \ldots, x_5)$ equal to the sum of the elements of $\overline{X}$ where each of the elements of $\overline{X}$ must be taken from a different row and column of a 5x5 matrix. For the first step, compute the sum of the 5 least elements in the matrix as the lower bound for all possible solution

vectors. For the second step, select an element of the matrix to be $x_1$. The set of solution vectors is now partitioned into the set of vectors which have $x_1 = a_{ij}$, for some particular i and j, and the complement of that set. The lower bounds must be calculated for each of the two subsets of the partition.

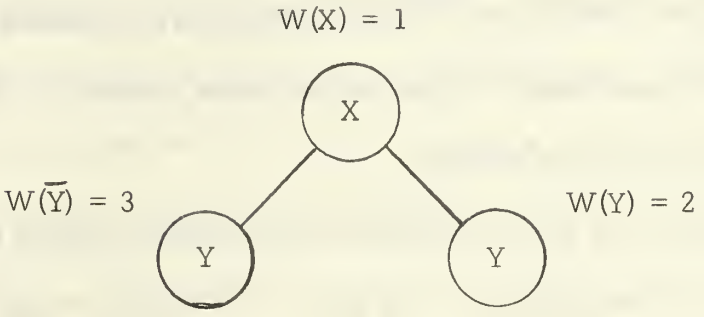
In general for the branch-and-bound search procedure, further partitions, "branching", and tightened bounds are made on the subset which has the least lower bound. After sufficient partitions some subset will contain but a single vector $(x_1^*, x_2^*, \ldots, x_n^*)$ and the bound of this subset will be the value of the objective function $z^* = f(x_1^*, x_2^*, \ldots, x_n^*)$. Any of the subsets which have lower bounds greater than $z^*$ cannot contain the optimal solution and may be disregarded in the search for the optimal vector. If $z^*$ is less than or equal to all the lower bounds on other subsets, $(x_1^*, x_2^*, \ldots, x_3^*)$ is an optimal solution. If $z^*$ is greater than the lower bound for some subset of the partitioned selection space, that subset must be further partitioned until the process of tightening the lower bound proves no optimal solution vectors are present in that subset, or until an optimal solution vector is isolated.

## B. BRANCH-AND-BOUND NOTATION

The following notation is used in the literature with respect to the branch-and-bound method [Refs. 1, 5, and 6].

| | |
|---|---|
| X | a subset of all feasible solution vectors |
| Y | a subset of X |
| $\overline{Y}$ | the complement of Y with respect to X |
| W(X) | a bound on the objective function for all possible solution vectors in X |

The subsets are usually represented as the nodes in a tree of nodes and branches with each node labeled by its associated bound.




Figures similar to this one give the technique its name of branch-and-bound.

## III. THE ALGORITHM

### A. BRANCH-AND-BOUND NOTATION FOR THE ROUTING PROBLEM

The basic method of the algorithm is to break up the set of all possible routes through n nodes into smaller and smaller subsets and to calculate for each of the subsets a lower bound on the cost of the best route in that subset. The bounds obtained are used as guides for the selection of further partitions into smaller subsets. The algorithm eventually isolates one or more subsets which are complete routes whose costs are less than or equal to the lower bounds for all other subsets. Such routes are optimal.

Arc (i,j) has the usual definition from network theory [Ref. 4] --- i.e., a directed path from node i to node j. The term "leg" refers to one of the sequence of arcs which form a complete route. The $k^{th}$ leg of a route between n cities is that arc (i,j) which is traversed between the $k^{th}$ and $(k+1)^{st}$ cities visited in sequence on the route. The route between n cities has (n-1) legs.

As described in Section II, the algorithm generates a tree whose nodes represent subsets of routes. The notation for the nodes of the tree is:

(i,j;k) is a subset of routes which have arc (i,j) for the $k^{th}$ leg, and

$(\overline{i,j};k)$ is the complement of (i,j;k).

Matrix reduction is the process of subtracting the minimum element in a matrix from each element in the matrix so that the least element in the reduced matrix is zero. The amount subtracted from each element is called the reducing constant. A set of matrices is called "reduced" if the least element in each matrix is zero.

"Restriction" of the set of cost matrices is the process of deleting the matrices and arcs which are no longer possible for the subset of routes being considered.

Arcs and legs are said to have been "committed" after they have been chosen to define a subset of possible routes. When (n-1) arcs have been committed to (n-1) legs, a route is completely determined.

Other notation used in the algorithm is listed below.

$R_k, (r_{ij}^k)$ the matrix of costs of traversing arc (i,j) on the $k^{th}$ leg of the route.

$A_k, (a_{ij})$ a matrix with elements that are the distances between pairs of nodes.

$c_k$ a multiplier (cost per unit distance) that is a function of the $k^{th}$ sequence position of a route.

$g$ $\sum r_{ij}^k$ summed over the set of (i,j;k) for committed arcs and legs.

$M_k, (m_{ij}^k)$ the current matrix of costs of traversing arc (i,j) on the $k^{th}$ leg of the route. Initially $M_k = R_k$ but is changed by the operation of the algorithm.

$M_k'$ the reduced form of $M_k$.

$q(i_k, j_k; k)$ the reducing constant for $M_k$.

$q(i_k, j_k; k) = \min_i \min_j m_{ij}^k$.

x represents plus infinity as a matrix element. x is used as an element in a cost matrix to block the path between the associated node pair.

W(X) the lower bound label attached to the tree for node X.

$\theta(i_k, j_k; k)$ the second smallest element in $M_k'$.

$\theta(i_k, j_k; k) = \min_{ij \neq i_k j_k} m_{ij}'^k$.

$\theta(i_o, j_o; k_o) = \max_k \theta(i_k, j_k; k)$.

## B. DETAILS OF THE BASIC ALGORITHM

The algorithm, which is listed here in complete detail, uses the branch-and-bound technique to find the least cost route between n nodes when the matrix of costs changes for each leg in sequence on the route. It is assumed that the set of matrices $R_k$ can be specified for all (n-1) legs of the route. The range of values for the elements of $R_k$ is not restricted.

A simplified flow chart of the algorithm is given in Section III.B.2.

### 1. The Steps of the Basic Algorithm

Step 1

The initial set up of the algorithm is made as follows.

1. Set $M_k = R_k$ for $k = 1,2,\ldots,n$.

2. X is the set of all possible routes.

3. Set $Z_o$ equal to plus infinity. $Z_o$ will be the cost of the optimal route at the end of the algorithm.

Step 2

1. For each $k = 1,2,\ldots,n-1$, find $i_k, j_k$, and $q(i_k, j_k; k)$ such that $q(i_k, j_k; k) = \min_i \min_j m_{ij}^k$.

2. Reduce $M_k$ to $M_k'$ where $m_{ij}'^k = m_{ij}^k - q(i_k, j_k; k)$ for all $i, j, k$.

3. Label node X with $W(X) = \sum q(i_k, j_k; k)$ summed over $k = 1,2,\ldots,n-1$.

Step 3

Choose the subset for the next tree extension by the following procedure.

1. $\theta(i_k, j_k; k) = \min_{ij \neq i_k j_k} m_{ij}'^k$ for all k of uncommitted legs. $\theta(i_k, j_k; k)$ is infinity if $M_k'$ is a single element.

2. $\theta(i_o, j_o; k_o) = \max_k \theta(i_k, j_k; k)$.

3. Then $Y = (i_o, j_o; k_o)$ and $\overline{Y} = (\overline{i_o, j_o}; k_o)$ are the next branches from X.

Step 4

Label $\overline{Y}$ by $W(\overline{Y}) = W(X) + \theta(i_o, j_o; k_o)$.

Step 5

1. Delete $M'_{k_o}$.

2. a. Delete all elements in $M'_{k_o+1}$ except row $j_o$.

   b. Delete columns $i_o$ and $j_o$ in $M'_{k_o+1}$.

3. a. Delete all elements in $M'_{k_o-1}$ except column $i_o$.

   b. Delete rows $i_o$ and $j_o$ in $M'_{k_o-1}$.

4. Delete rows $i_o$ and $j_o$ and columns $i_o$ and $j_o$ in all $M'_k$ except for $k = k_o - 1$ and $k_o + 1$.

5. Relabel the matrices as $M_k$.

6. Leg $k_o$ and arc$(i_o, j_o)$ are now committed.

Step 6

1. For each k, where leg k has not yet been committed to a route, find $i_k, j_k$ and $q(i_k, j_k; k)$ such that $q(i_k, j_k; k) = \min_i \min_j m^k_{ij}$.

2. Reduce $M_k$ to $M'_k$ where $m'^k_{ij} = m^k_{ij} - q(i_k, j_k; k)$ for all i, j, k of uncommitted arcs.

3. Label Y by $W(Y) = W(X) + \sum q(i_k, j_k; k)$ summed over k for uncommitted legs.

Step 7

If (n-2) legs of the route have been committed go to Step 10. Otherwise go to Step 8.

Step 8

1. Select the node X from which to branch by choosing the terminal node which has the smallest label W(X).

2. If $Z_o$ is less than or equal to W(X), the optimal route has been found. STOP.

3. If X = Y of Step 6 go to Step 3. Otherwise go to Step 9.

Step 9

Set up the cost matrices and label node X as follows.

1. Compute $g = \sum r_{ij}^{k}$ summed over the set of (i,j;k) for committed arcs and legs.

2. If no legs have been committed, set $M_k = R_k$, otherwise set $M_k' = R_k$.

3. Carry out substeps 1 through 5 of Step 5 for each of the committed arcs and legs.

4. Block paths which are not allowed.

5. Carry out Step 6 substeps 1 and 2.

6. Label X with $W(X) = g + q(i_k, j_k; k)$ summed over k for the uncommitted legs.

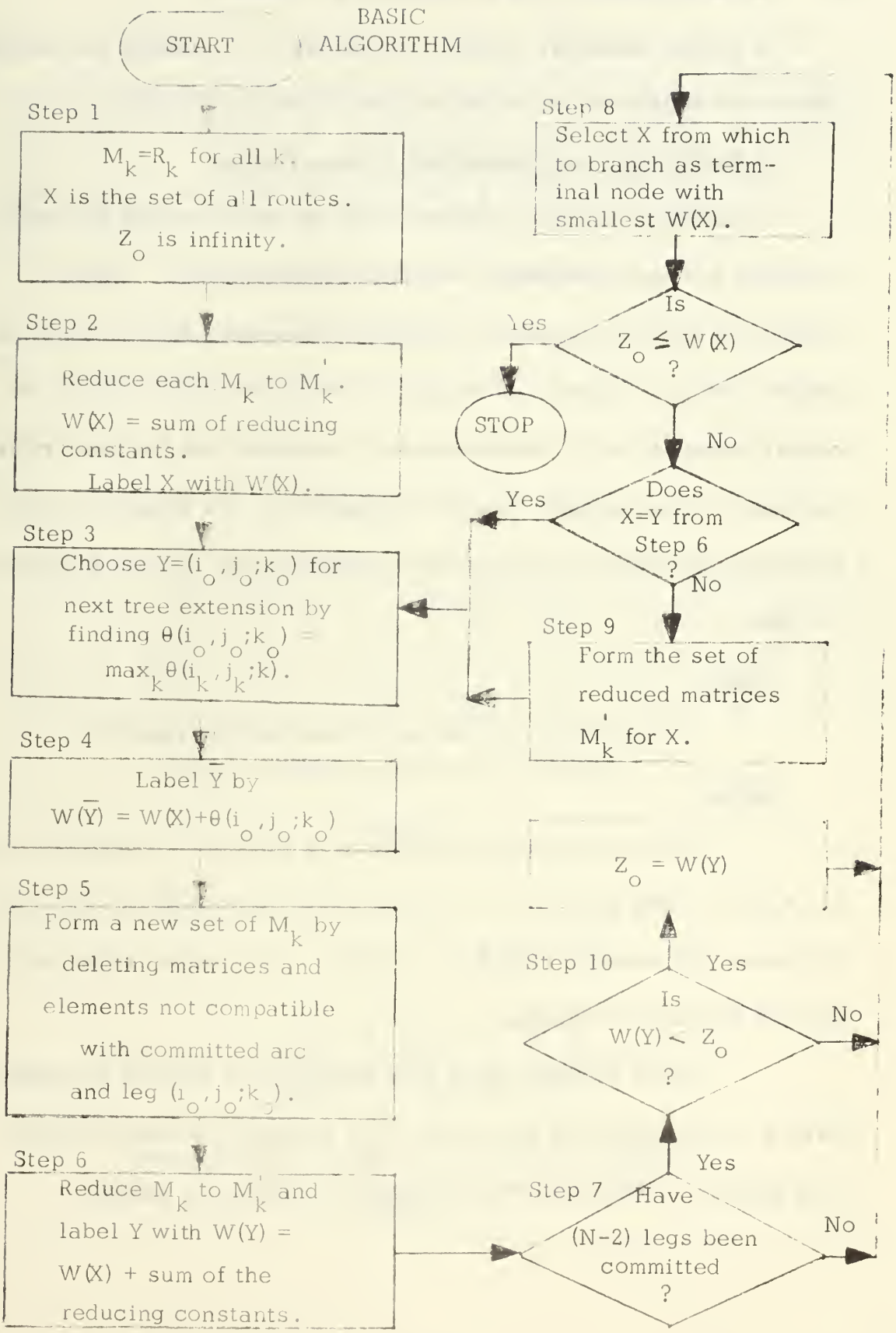
7. Go to Step 3.

Step 10

If W(Y) is less than $Z_o$, set $Z_o = W(Y)$. Go to Step 8.

2. Flow Chart of the Basic Algorithm

The flow chart of the basic algorithm is given on the next page.


START
BASIC ALGORITHM
Step 1
$M_k = R_k$ for all k.
X is the set of all routes.
$Z_o$ is infinity.
Step 2
Reduce each $M_k$ to $M_k'$.
W(X) = sum of reducing constants.
Label X with W(X).
Step 3
Choose $Y=(i_o, j_o; k_o)$ for next tree extension by finding $\theta(i_o, j_o; k_o) = \max_k \theta(i_k, j_k; k)$.
Step 4
Label Y by
$W(Y) = W(X) + \theta(i_o, j_o; k_o)$
Step 5
Form a new set of $M_k$ by deleting matrices and elements not compatible with committed arc and leg $(i_o, j_o; k_o)$.
Step 6
Reduce $M_k$ to $M_k'$ and label Y with W(Y) = W(X) + sum of the reducing constants.
Step 7
Have (N-2) legs been committed ?
Yes
No
Step 8
Select X from which to branch as terminal node with smallest W(X).
Is $Z_o \leq W(X)$ ?
Yes
STOP
No
Does X=Y from Step 6 ?
Yes
No
Step 9
Form the set of reduced matrices $M_k'$ for X.
$Z_o = W(Y)$
Step 10
Is $W(Y) < Z_o$ ?
Yes
No

## C. EXPLANATION OF THE ALGORITHM BY EXAMPLE

A simple numerical example will be used to illustrate the algorithm while tracing through the steps detailed in Section III. B.1.

A Simple Sequence-Dependent Routing Problem

Suppose an itinerant salesman must be routed so that his travel expenses are minimized while visiting 5 different cities. He must complete a leg of his route on each of 4 consecutive days. Travel expenses vary as a function of the day on which the travel occurs. At certain times no public transportation is available and the costs reflect the price of the available charter transportation. All possible costs have been tabulated for each of the 4 traveling days and are presented in Figure 1.

Step 1

The set of $M_k$ are the set of matrices of Figure 1.

Step 2

The set of reducing constants is $q(1,2;1) = 3$, $q(2,3;2) = 5$, $q(5,4;3) = 2$, and $q(4,5;4) = 1$. The set of $M_k$ is reduced to the set of $M_k'$ shown in Figure 2. $W(X) = 3 + 5 + 2 + 1 = 11$, which is the sum of the four reducing constants.

Since the route must have four legs, it must as a minimum have a cost equal to the sum of the least elements in each of the four cost matrices. Therefore W(X) is indeed a valid lower bound.

| | $M_1$ | | | | | $M_2$ | | | | | $M_3$ | | | | | $M_4$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| 1 | X | 3 | 11 | 14 | 6 | X | 6 | 11 | 12 | 7 | X | 6 | 14 | 9 | 29 | X | 17 | 11 | 22 | 9 |
| 2 | 10 | X | 7 | 9 | 15 | 13 | X | 5 | 10 | 13 | 16 | X | 24 | 8 | 15 | 28 | X | 16 | 19 | 10 |
| 3 | 23 | 12 | X | 29 | 4 | 26 | 24 | X | 15 | 14 | 7 | 25 | X | 3 | 17 | 24 | 20 | X | 21 | 6 |
| 4 | 22 | 24 | 13 | X | 5 | 21 | 8 | 20 | X | 18 | 5 | 18 | 15 | X | 13 | 15 | 14 | 12 | X | 1 |
| 5 | 16 | 19 | 20 | 26 | X | 9 | 16 | 23 | 29 | X | 26 | 12 | 23 | 2 | X | 14 | 16 | 7 | 13 | X |

The initial set of cost matrices.

Figure 1

| | $M_1'$ | | | | | $M_2'$ | | | | | $M_3'$ | | | | | $M_4'$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| 1 | X | 0 | 8 | 11 | 3 | X | 1 | 6 | 7 | 2 | X | 4 | 12 | 7 | 27 | X | 16 | 10 | 21 | 8 |
| 2 | 7 | X | 4 | 6 | 12 | 8 | X | 0 | 5 | 8 | 14 | X | 22 | 6 | 13 | 27 | X | 15 | 18 | 9 |
| 3 | 20 | 9 | X | 26 | 1 | 21 | 19 | X | 10 | 9 | 5 | 23 | X | 1 | 15 | 23 | 19 | X | 20 | 5 |
| 4 | 19 | 21 | 10 | X | 2 | 16 | 3 | 15 | X | 13 | 3 | 16 | 13 | X | 11 | 14 | 13 | 11 | X | 0 |
| 5 | 13 | 16 | 17 | 23 | X | 4 | 11 | 18 | 24 | X | 24 | 10 | 21 | 0 | X | 13 | 15 | 6 | 12 | X |

Reduced matrices at the end of Step 2.

Figure 2

| | $M_1$ | | | $M_2$ | | | $M_3$ |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 | 4 |
| 1 | X | 0 | 8 | X | 1 | 6 | 7 |
| 2 | 7 | X | 4 | 8 | X | 0 | 6 |
| 3 | 20 | 9 | X | 21 | 19 | X | 1 |

Restricted set of matrices at the end of Step 5.

Figure 3

Step 3

$\theta(i_k, j_k; k) = 1, 1, 1, 5$ for $k = 1, 2, 3, 4$ respectively. Hence $\theta(i_o, j_o; k_o) = \theta(4, 5; 4) = 5$ and the two subsets $(4, 5; 4)$ and $(\overline{4, 5}; 4)$ are chosen for the next extension of the tree.

Step 3 partitions X into subsets Y and $\overline{Y}$ by choosing the arc and leg on which to base the next branching. The object is to include in the route the arc and leg which is judged most likely to be on the optimal route. It is argued that one of the zero elements is a likely candidate to be on the optimal route; and that arc and leg are chosen which have $m'^{k}_{ij}$ equal zero and which, if not chosen, cause the greatest possible alternative cost. The arc and leg so chosen will raise the bound on $\overline{Y}$ the most. For each reduced matrix $M'_k$ a $\theta(i_k, j_k; k)$ equal to the second smallest element in the matrix is calculated. $\theta(i_k, j_k; k) = 0$ if $M'_k$ has two or more zero elements. The maximum over k of the $\theta(i_k, j_k; k)$ is the highest cost for not placing the zero cost arc on leg $k_o$ of the route.

Step 4

$$W(\overline{Y}) = W(X) + \theta(4, 5; 4) = 16.$$

Step 5

The new restricted set of $M_k$ is formed by eliminating arcs in the $M'_k$ matrices which are not possible for subsets of routes in Y. Since the subset $(4, 5; 4)$ was chosen, the restricted set of $M_k$ shown in Figure 3 are formed as follows.

1. $M'_4$ is deleted.

2. All of $M'_3$ except column 4 is deleted, and $m'^3_{44}$ and $m'^3_{54}$ are also deleted.

3. Rows and columns 4 and 5 are deleted from $M'_1$ and $M'_2$.

The restrictions brought about by the choice of arc (4,5) for leg 4 are now satisfied. Leg 3 must end at city 4 and no other legs are allowed to start or end at cities 4 and 5. It should be clear that if leg 4 were not the last leg in the route, in accordance with the algorithm, leg 5 would be restricted to begin at city 5 and the restricted form of $M'_5$ would have elements only in row 5.

Step 6

The set of $q(i_k, j_k; k) = 0, 0, 1$ for $k = 1, 2, 3$ respectively. Hence $W(Y) = W(X) + 1 = 12$. The restrictions introduced in Step 5 make it impossible to use the zero cost arc in the $M'_3$ matrix at the end of Step 2. The lower bound is therefore tightened by the amount of the least element in the current $M_3$. Figure 4 shows the set of reduced matrices and Figure 5 shows the labeled tree that has been constructed up to this point.

Step 7

Only one leg has been committed, so Step 8 is next.

Step 8

Node (4,5;4) has the smallest lower bound and is chosen for the next X from which to branch. $Z_o$ remains infinity. Since X is

| $M_1'$ | | | $M_2'$ | | | $M_3'$ |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 | 4 |
| 1 | X | 0 | 8 | X | 1 | 6 | 6 |
| 2 | 7 | X | 4 | 8 | X | 0 | 5 |
| 3 | 20 | 9 | X | 21 | 19 | X | 0 |

Reduced matrices at the end of Step 6.

Figure 4

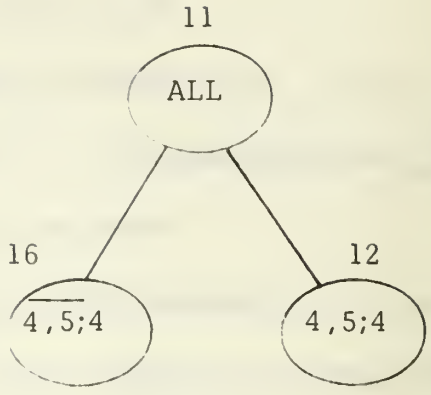



Figure 5

| $M_1'$ | | | $M_2'$ |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | X | 0 | 6 |
| 2 | 7 | X | 0 |

Figure 6

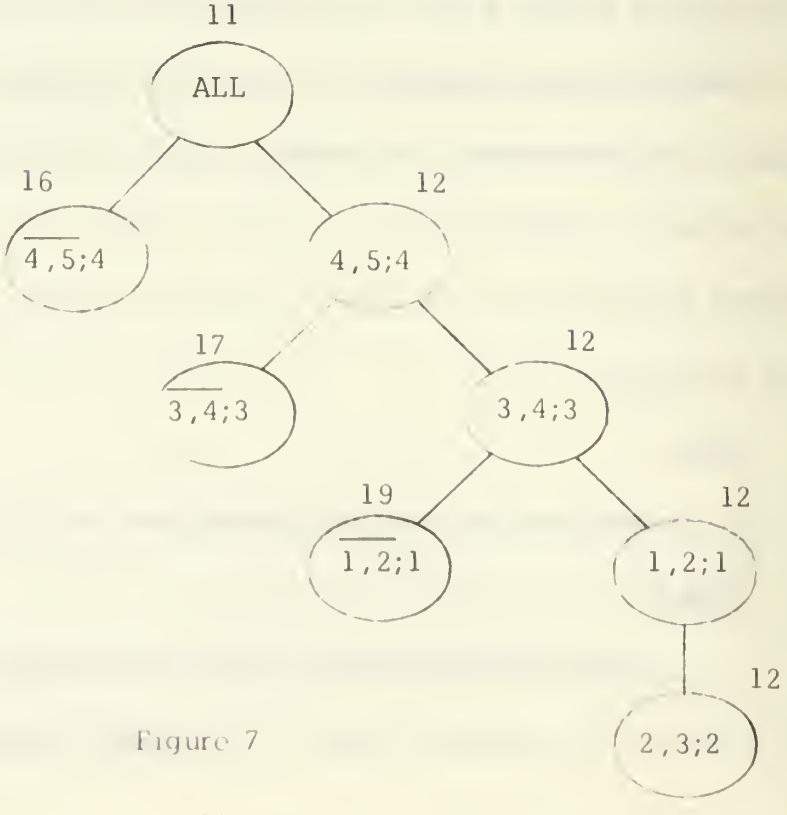



Figure 7

the same as the last Y, Step 3 is executed next. Step 9 is bypassed in this case since the correct set of $M_k$ is available from the end of Step 6.

Second Iteration

Based on the set of matrices in Figure 4, $\theta(i_o, j_o; k_o) = \theta(3,4;3)$ in Step 3 and the tree is extended to $(3,4;3)$ and $(\overline{3,4};3)$. In Step 4 $W(\overline{Y}) = 12 + 5 = 17$. After Step 6 the set of $M_k'$ is that of Figure 6. Since all the reducing constants are zero, $(3,4;3)$ is labeled with $W(X) + 0 = 12$. In Step 8 $(3,4;3)$ is selected to be X and the third iteration begins with the execution of Step 3.

Third Iteration

In Step 3 Y is chosen to be $(1,2;1)$ since $\theta(i_o, j_o; k_o) = \theta(1,2;1) = 7$. Therefore $(\overline{1,2};1)$ is labeled with 19. After Step 5 only the single element for arc $(2,3)$ in leg 2 remains. A complete route is now isolated since leg 2 with only one possible element must be included. After Step 6 the cost of leg 2 is incorporated in the label of $(1,2;1)$ which is now also the total cost of the route. Steps 7 and 10 are executed so that $Z_o = 12$, the label on $(1,2;1)$. Finally, since no terminal node has a label less than 12, the algorithm is stopped in Step 8.

The optimal route is $(1,2)$, $(2,3)$, $(3,4)$, $(4,5)$ at a cost of 12. The completed tree for the problem is shown in Figure 7. Whereas complete enumeration would have required the calculation of the costs of 120 different possible combinations, branch-and-bound finds the optimal solution by calculating the complete cost of but a single route

and the lower bound on three other sets of routes. This clearly demonstrates the powerful potential for efficiency of the branch-and-bound technique.

If the calculation of a label for a node in branch-and-bound is considered comparable to the evaluation of the recursive equation for one value of a state variable in dynamic programming, an approximate comparison of efficiencies between branch-and-bound and dynamic programming may be made. The dynamic programming solution to the problem just solved would have required evaluation of the recursive equation for 30 different values of the state variable and 30 comparisons to select the optimal route. The branch-and-bound solution requires the calculation of 7 node labels and 15 comparisons for both the selection of the X nodes from which to branch and the selection of the Y nodes for tree extension. Consequently, for this particular problem, the branch-and-bound algorithm is the more efficient solution procedure.

## D. A CONSTANT DISTANCE AND VARIABLE COST RATE ($C_k$) PROBLEM

A problem of some interest is the minimum cost route when the cost for each leg is the product of distance traveled and a sequence dependent cost factor. The branch-and-bound algorithm is used to solve this type of problem in the example below.

### A Constant Distance, Variable Cost Rate Example

The optimal route passing through 6 cities is desired. The cost of travel between a given pair of cities is $c_k a_{ij}$. $c_k$ is a function of the

Daily Cost Rates

| Day (k) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $C_k$ | 4 | 1 | 3 | 2 | 5 |

Figure 8

Distance Matrix A

| From \ To | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 10 | 6 | 8 | 7 | 3 |
| 2 | 9 | X | 5 | 9 | 1 | 8 |
| 3 | 6 | 8 | X | 2 | 4 | 4 |
| 4 | 3 | 5 | 5 | X | 6 | 4 |
| 5 | 2 | 8 | 3 | 2 | X | 2 |
| 6 | 4 | 6 | 4 | 5 | 3 | X |

Figure 9



$R_1$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 40 | 24 | 32 | 28 | 12 |
| 2 | 36 | X | 20 | 36 | 4 | 32 |
| 3 | 24 | 32 | X | 8 | 16 | 16 |
| 4 | 12 | 20 | 20 | X | 24 | 16 |
| 5 | 8 | 32 | 12 | 8 | X | 8 |
| 6 | 16 | 24 | 16 | 20 | 12 | X |

$R_2$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 10 | 6 | 8 | 7 | 3 |
| 2 | 9 | X | 5 | 9 | 1 | 8 |
| 3 | 6 | 8 | X | 2 | 4 | 4 |
| 4 | 3 | 5 | 5 | X | 6 | 4 |
| 5 | 2 | 8 | 3 | 2 | X | 2 |
| 6 | 4 | 6 | 4 | 5 | 3 | X |

$R_3$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 30 | 18 | 24 | 21 | 9 |
| 2 | 27 | X | 15 | 27 | 3 | 24 |
| 3 | 18 | 24 | X | 6 | 12 | 12 |
| 4 | 9 | 15 | 15 | X | 18 | 12 |
| 5 | 6 | 24 | 9 | 6 | X | 6 |
| 6 | 12 | 18 | 12 | 15 | 9 | X |

$R_4$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 20 | 12 | 16 | 14 | 6 |
| 2 | 18 | X | 10 | 18 | 2 | 16 |
| 3 | 12 | 16 | X | 4 | 8 | 8 |
| 4 | 6 | 10 | 10 | X | 12 | 8 |
| 5 | 4 | 16 | 6 | 4 | X | 4 |
| 6 | 8 | 12 | 8 | 10 | 6 | X |

$R_5$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 50 | 30 | 40 | 35 | 15 |
| 2 | 45 | X | 25 | 45 | 5 | 40 |
| 3 | 30 | 40 | X | 10 | 20 | 20 |
| 4 | 15 | 25 | 25 | X | 30 | 20 |
| 5 | 10 | 40 | 15 | 10 | X | 10 |
| 6 | 20 | 30 | 20 | 25 | 12 | X |

Initial set of cost matrices.

Figure 10

day on which travel is performed and is called the daily cost rate. It is assumed that each leg of the route will be traversed on a different single day. Five days are required for travel. The cost rate (units of cost per unit distance) for each day is given in Figure 8 and the distance matrix in Figure 9.

The set of (n-1) cost matrices, where $R_k = c_k(a_{ij})$, is formed and shown in Figure 10. The set of reduced matrices $M_k$ after Step 2 of the algorithm is shown in Figure 11. After the first iteration of the algorithm is completed through Step 6, the labeled tree is that shown in Figure 12. Since the bound on the left hand node is less than the bound on the right hand node, Step 9 of the algorithm is executed. Arc (2,5) is blocked for leg 5 by setting $m_{25}^5$ to plus infinity. The set of matrices is reduced and shown in Figure 13. The sum of the reducing constants is found to be 20. Since g is zero (no arcs have been committed in this subset), the label for the node is 20, which agrees with that determined on the first iteration for $\overline{Y}$.

The completed tree for this example is shown in Figure 14. Only one complete route is enumerated by the algorithm, the optimal route (2,5), (5,1), (1,6), (6,3), (3,4). It is noteworthy that four consecutive nodes in the left most branch of the tree have the same lower bound. Examination of the reduced matrices for $(\overline{2,5};2)$ reveals the reason. $\theta(i_k, j_k;k)$ in each matrix of Figure 15 is zero. All of the $\theta(i_k, j_k;k)$ remain zero in later branches until sufficient arcs have become blocked to make at least one $\theta(i_k, j_k;k)$ different from zero.

$M_1'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 36 | 20 | 28 | 24 | 8 |
| 2 | 32 | X | 16 | 32 | 0 | 28 |
| 3 | 20 | 28 | X | 4 | 12 | 12 |
| 4 | 8 | 16 | 16 | X | 20 | 12 |
| 5 | 4 | 28 | 8 | 4 | X | 4 |
| 6 | 12 | 20 | 12 | 16 | 8 | X |

$M_2'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 9 | 5 | 7 | 6 | 2 |
| 2 | 8 | X | 4 | 8 | 0 | 7 |
| 3 | 5 | 7 | X | 1 | 3 | 3 |
| 4 | 2 | 4 | 4 | X | 5 | 3 |
| 5 | 1 | 7 | 2 | 1 | X | 1 |
| 6 | 3 | 5 | 3 | 4 | 2 | X |

$M_3'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 27 | 15 | 21 | 18 | 6 |
| 2 | 24 | X | 12 | 24 | 0 | 21 |
| 3 | 15 | 21 | X | 3 | 9 | 9 |
| 4 | 6 | 12 | 12 | X | 15 | 9 |
| 5 | 3 | 21 | 6 | 3 | X | 3 |
| 6 | 9 | 15 | 9 | 12 | 6 | X |

$M_4'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 18 | 10 | 14 | 12 | 4 |
| 2 | 16 | X | 8 | 16 | 0 | 14 |
| 3 | 10 | 14 | X | 2 | 6 | 6 |
| 4 | 4 | 8 | 8 | X | 10 | 6 |
| 5 | 2 | 14 | 4 | 2 | X | 2 |
| 6 | 6 | 10 | 6 | 8 | 4 | X |

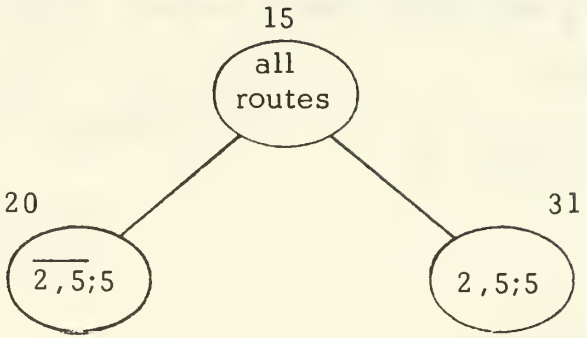
$M_5'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 45 | 25 | 35 | 30 | 10 |
| 2 | 40 | X | 20 | 40 | 0 | 35 |
| 3 | 25 | 35 | X | 5 | 15 | 15 |
| 4 | 10 | 20 | 20 | X | 25 | 15 |
| 5 | 5 | 35 | 10 | 5 | X | 5 |
| 6 | 15 | 25 | 15 | 20 | 10 | X |

Reduced matrices after Step 2.

Figure 11






Labeled tree at end of first iteration.

Figure 12

$M_1'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 36 | 20 | 28 | 24 | 8 |
| 2 | 32 | X | 16 | 32 | 0 | 28 |
| 3 | 20 | 28 | X | 4 | 12 | 12 |
| 4 | 8 | 16 | 16 | X | 20 | 12 |
| 5 | 4 | 28 | 8 | 4 | X | 4 |
| 6 | 12 | 20 | 12 | 16 | 8 | X |

$M_2'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 9 | 5 | 7 | 6 | 2 |
| 2 | 8 | X | 4 | 8 | 0 | 7 |
| 3 | 5 | 7 | X | 1 | 3 | 3 |
| 4 | 2 | 4 | 4 | X | 5 | 3 |
| 5 | 1 | 7 | 2 | 1 | X | 1 |
| 6 | 3 | 5 | 3 | 4 | 2 | X |

$M_3'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 27 | 15 | 21 | 18 | 6 |
| 2 | 24 | X | 12 | 24 | 0 | 21 |
| 3 | 15 | 21 | X | 3 | 9 | 9 |
| 4 | 6 | 12 | 12 | X | 15 | 9 |
| 5 | 3 | 21 | 6 | 3 | X | 3 |
| 6 | 9 | 15 | 9 | 12 | 6 | X |

$M_4'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 18 | 10 | 14 | 12 | 4 |
| 2 | 16 | X | 8 | 16 | 0 | 14 |
| 3 | 10 | 14 | X | 2 | 6 | 6 |
| 4 | 4 | 8 | 8 | X | 10 | 6 |
| 5 | 2 | 14 | 4 | 2 | X | 2 |
| 6 | 6 | 10 | 6 | 8 | 4 | X |

$M_5'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 40 | 20 | 30 | 25 | 5 |
| 2 | 35 | X | 15 | 30 | X | 30 |
| 3 | 20 | 30 | X | 0 | 10 | 10 |
| 4 | 5 | 15 | 15 | X | 20 | 10 |
| 5 | 0 | 30 | 5 | 0 | X | 0 |
| 6 | 10 | 20 | 10 | 15 | 5 | X |

Reduced matrices after Step 9.

Figure 13

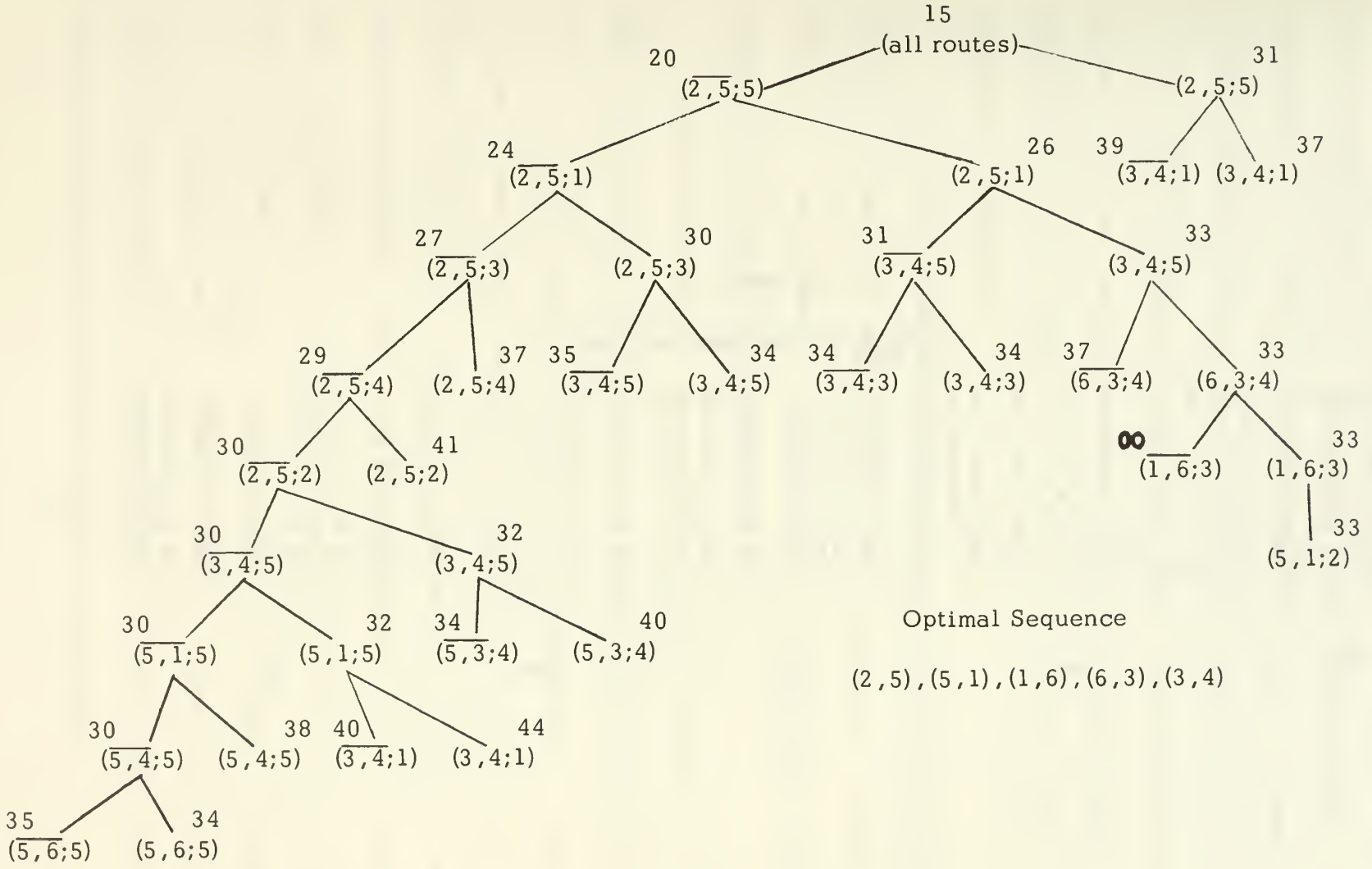

15
(all routes)
20
(2,5;5)
31
(2,5;5)
24
(2,5;1)
26
(2,5;1)
39
(3,4;1)
37
(3,4;1)
27
(2,5;3)
30
(2,5;3)
31
(3,4;5)
33
(3,4;5)
29
(2,5;4)
37
(2,5;4)
35
(3,4;5)
34
(3,4;5)
34
(3,4;3)
34
(3,4;3)
37
(6,3;4)
33
(6,3;4)
30
(2,5;2)
41
(2,5;2)
∞
(1,6;3)
33
(1,6;3)
30
(3,4;5)
32
(3,4;5)
33
(5,1;2)
30
(5,1;5)
32
(5,1;5)
34
(5,3;4)
40
(5,3;4)
30
(5,4;5)
38
(5,4;5)
40
(3,4;1)
44
(3,4;1)
35
(5,6;5)
34
(5,6;5)
Optimal Sequence
(2,5),(5,1),(1,6),(6,3),(3,4)

Figure 14

$M_1'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 32 | 16 | 24 | 20 | 4 |
| 2 | 28 | X | 12 | 28 | X | 24 |
| 3 | 16 | 24 | X | 0 | 8 | 8 |
| 4 | 4 | 12 | 12 | X | 16 | 8 |
| 5 | 0 | 24 | 4 | 0 | X | 0 |
| 6 | 8 | 16 | 8 | 12 | 4 | X |

$M_2'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 8 | 4 | 6 | 5 | 1 |
| 2 | 7 | X | 3 | 7 | X | 6 |
| 3 | 4 | 6 | X | 0 | 2 | 2 |
| 4 | 1 | 3 | 3 | X | 4 | 2 |
| 5 | 0 | 6 | 1 | 0 | X | 0 |
| 6 | 2 | 4 | 2 | 3 | 1 | X |

$M_3'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 24 | 12 | 18 | 15 | 3 |
| 2 | 21 | X | 9 | 21 | X | 18 |
| 3 | 12 | 18 | X | 0 | 6 | 6 |
| 4 | 3 | 9 | 9 | X | 12 | 6 |
| 5 | 0 | 18 | 3 | 0 | X | 0 |
| 6 | 6 | 12 | 6 | 9 | 3 | X |

$M_4'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 16 | 8 | 12 | 10 | 2 |
| 2 | 14 | X | 6 | 14 | X | 12 |
| 3 | 8 | 12 | X | 0 | 4 | 4 |
| 4 | 2 | 6 | 6 | X | 8 | 4 |
| 5 | 0 | 12 | 2 | 0 | X | 0 |
| 6 | 4 | 8 | 4 | 6 | 2 | X |

$M_5'$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | X | 40 | 20 | 30 | 25 | 5 |
| 2 | 35 | X | 15 | 35 | X | 30 |
| 3 | 20 | 30 | X | 0 | 10 | 10 |
| 4 | 5 | 15 | 15 | X | 20 | 10 |
| 5 | 0 | 30 | 5 | 0 | X | 0 |
| 6 | 10 | 20 | 10 | 15 | 5 | X |

Reduced matrices for $(\overline{2,5};2)$.

Figure 15

The question arises of what branching rule to use when the values of $\theta(i_k, j_k; k)$ are equal for two or more k. Of course, the best branching choice is that which causes as few other iterations as possible. But no rule that can be proved is known for making the choice. As shown in the tree, branching choices were made by taking zeroes in $M'_k$ in the order in which they appeared in the matrix. Leg 5 was chosen because $c_5$ was the highest cost rate multiplier and the route cost could be lessened by at least 5 units if one of the zero valued arcs in $M'_5$ could be placed on the route.

In any case no branching rule can eliminate the further branching required from (m-1) left hand nodes when m zeroes occur in a reduced matrix that is not symmetric. In Section III. E. the algorithm for the symmetric cost matrix case is discussed.

The branch-and-bound algorithm required the calculation of 33 labels and required the performance of 54 comparisons in order to choose Y nodes and 120 comparisons to find the X nodes from which to branch. For complete enumeration the cost of 720 different routes would be computed and compared. Dynamic programming would require 62 evaluations of the recursive equation and 62 comparisons to find the optimal route. The structure of the problem solved here causes numerous branches and increases the number of nodes that must be considered in the search for the next X node from which to branch further. Consequently for problems which generate numerous branches, branch-and-bound solutions may be inferior in efficiency to dynamic programming.

$R_1$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | X | 3 | 11 | 14 | 6 |
| 2 | 3 | X | 7 | 9 | 15 |
| 3 | 11 | 7 | X | 29 | 4 |
| 4 | 14 | 9 | 29 | X | 5 |
| 5 | 6 | 15 | 4 | 5 | X |

$R_2$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | X | 6 | 11 | 12 | 7 |
| 2 | 6 | X | 5 | 10 | 13 |
| 3 | 11 | 5 | X | 15 | 14 |
| 4 | 12 | 10 | 15 | X | 18 |
| 5 | 7 | 13 | 14 | 18 | X |

$R_3$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | X | 3 | 14 | 9 | 29 |
| 2 | 3 | X | 24 | 8 | 15 |
| 3 | 14 | 24 | X | 7 | 17 |
| 4 | 9 | 8 | 7 | X | 5 |
| 5 | 29 | 15 | 17 | 5 | X |

$R_4$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | X | 17 | 11 | 9 | 22 |
| 2 | 17 | X | 16 | 10 | 19 |
| 3 | 11 | 16 | X | 4 | 21 |
| 4 | 9 | 10 | 4 | X | 1 |
| 5 | 22 | 19 | 21 | 1 | X |

Figure 16

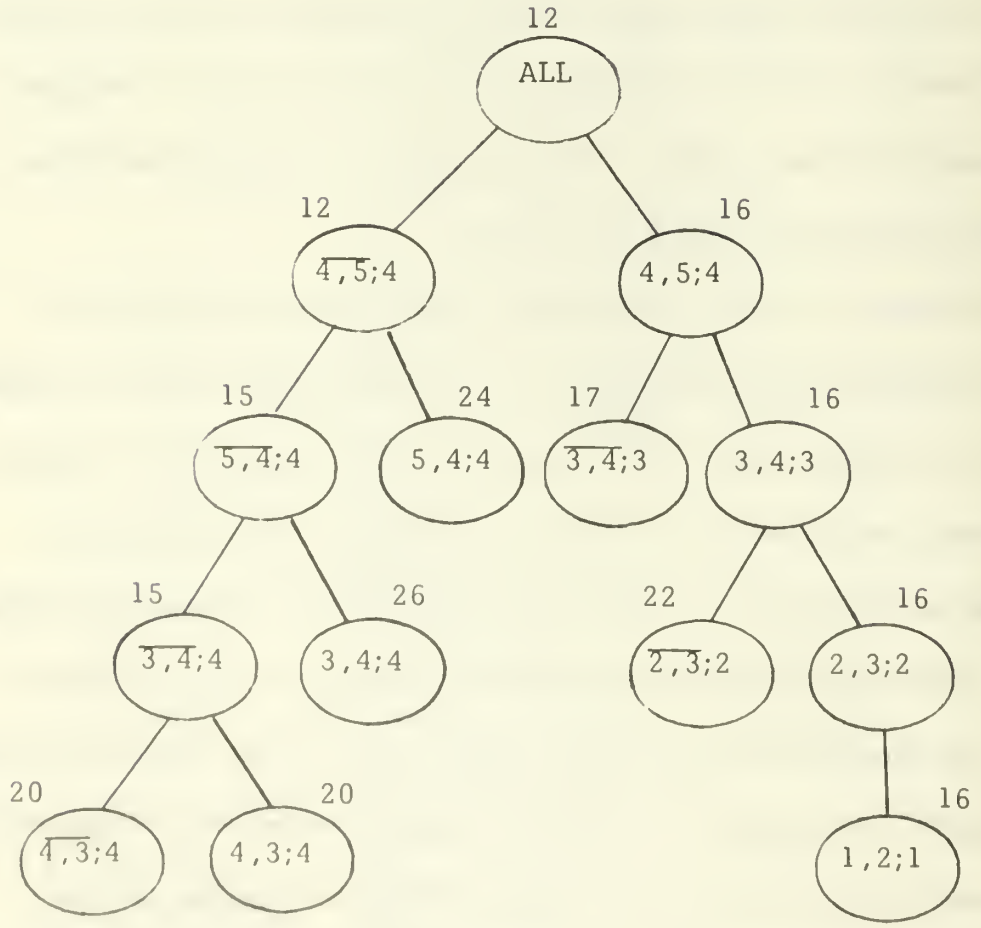



Figure 17

## E. SYMMETRIC COST MATRICES

### 1. Efficiency of Basic Algorithm for Symmetric Cost Matrices

The basic algorithm presented in Section III. B. will successfully solve the routing problem when all cost matrices are symmetric, but with poor efficiency. Symmetric cost matrices cause an increase in the number of branches because every reduced matrix will have two zero elements for the first iteration of the algorithm. The example in Section III. D. showed that multiple zeroes in the reduced cost matrices cause the branch-and-bound algorithm to be less efficient. A short example with symmetric cost matrices is given here to illustrate the difficulty inherent in symmetry for the basic algorithm.

Figure 16 is the set of four symmetric cost matrices for which a minimal cost route is to be found. Figure 17 is the solution tree which results from the use of the basic algorithm presented in Section III. B. Twice in the leftmost branch of Figure 17, two branchings occur before the lower bound is raised. This double branching caused by symmetry is avoided if a modification of the basic algorithm is used.

### 2. Modification to the Algorithm for Symmetric Matrices

In the modification of the algorithm, the arcs are considered undirected and the direction of the route is not specified until two adjacent legs are committed. The nodes of the tree are redefined as (i,j or j,i;k) with complementary node ($\overline{i,j}$ and $\overline{j,i};k$). The definition of a "directed" leg is necessary for the modified algorithm.

Leg k is said to be "directed (i,j)" if leg k is committed and the only possible direction for the route in leg k is from i to j.

Step 5A of the modified algorithm is used to specify direction for the set of directed legs. If leg $k_o$ is already directed, Step 5 of the basic algorithm is executed following Step 5B. Step 5 is more restrictive than Step 5C of the modification, in the sense that more matrix elements are deleted. Step 7 is changed in order that the direction of the last leg committed to the route can be determined. Step 9 is modified to account for the difference between "directed" and "undirected" legs.

Because of symmetry only upper triangles of the cost matrices are used in the modified algorithm. This results in a reduction of storage required by the amount $n(n^2 - 1)/2$.

The basic algorithm is modified for the symmetric cost matrices by inserting Step 5A, Step 5B, and Step 5C between Step 4 and Step 5 in the basic algorithm and by making changes to Step 7 and Step 9. The details of the modification are given immediately below.

Step 5A

1. If neither leg $(k_o + 1)$ nor leg $(k_o - 1)$ has been committed go to Step 5C, otherwise, go to substep 2.

2. Set $t = k_o, u = i_o, s = j_o$. Execute the first of substeps 3, 4, 5, and 6, which is applicable.

3. a. If leg (t + 1) has undirected arc (p,q) then

   (1) if s = p, leg t is directed (u,s) and leg (t + 1) is directed (p,q).

(2) if s = q, leg t is directed (u,s) and leg (t + 1) is directed (q,p).

(3) if u = p, leg t is directed (s,u) and leg (t + 1) is directed (p,q).

(4) if u = q, leg t is directed (s,u) and leg (t + 1) is directed (q,p).

b. Then if leg (t - 1) is committed and undirected, set t = t - 1 and execute substep 4 below. Otherwise go to Step 5B.

4. a. If leg (t + 1) is directed (p,q) and leg t has undirected arc (u,s) then

(1) if s = p, leg t is directed (u,s).

(2) if u = p, leg t is directed (s,u).

b. If leg (t - 1) is committed and undirected, set t = t - 1 and repeat substep 4. Otherwise go to Step 5B.

5. a. If leg (t - 1) has undirected arc (p,q) then

(1) if s = p, leg t is directed (s,u) and leg (t - 1) is directed (q,p).

(2) if s = q, leg t is directed (s,u) and leg (t - 1) is directed (p,q).

(3) if u = p, leg t is directed (u,s) and leg (t - 1) is directed (q,p).

(4) if u = q, leg t is directed (u,s) and leg (t - 1) is directed (p,q).

b. If leg (t + 1) is committed and undirected, set t = t + 1 and execute substep 6 below. Otherwise go to Step 5B.

6. a. If leg (t - 1) is directed (p,q) and leg t has undirected arc (u,s) then

(1) if s = q, leg t is directed (s,u).

(2) if u = q, leg t is directed (u,s).

b. If leg (t + 1) is committed and undirected, set t = t + 1 and repeat substep 6. Otherwise go to Step 5B.

Step 5B

If leg $k_o$ is directed (p,q) as a result of Step 5A, the ordered pair for Step 5 is $(i_o, j_o) = (p,q)$. Go to Step 5 in the basic algorithm.

Step 5C

1. Delete $M'_{k_o}$.

2. Delete all elements in $M'_{k_o+1}$ except rows $i_o$ and $j_o$.

3. Delete all elements in $M'_{k_o-1}$ except columns $i_o$ and $j_o$.

4. Delete $m'^{k}_{i_o j_o}$ or $m'^{k}_{j_o i_o}$, whichever is present in the upper triangle of the matrix, for $k = k_o - 1$ and $k = k_o + 1$.

5. Delete rows $i_o$ and $j_o$ and columns $i_o$ and $j_o$ in all matrices except $M'_{k_o+1}$ and $M'_{k_o-1}$.

6. Relabel the matrices as $M_k$.

7. Leg $k_o$ and arc $(i_o, j_o)$ are now committed.

8. Go to Step 6.

Changes to Step 7 and Step 9

1. In Step 7 change "(n - 2)" to "(n - 1)".

2. Replace substep 3 in Step 9 with

"3. Carry out Step 5, substeps 1 through 5, for each of the directed legs and Step 5C, substeps 1 through 6, for each of the committed but undirected legs."

3. An Example Using the Algorithm Modified for Symmetric Cost Matrices

Once again suppose a minimum cost route for the set of matrices in Figure 16 is to be found.

First Iteration

The set of reduced triangular $M'_k$ at the end of Step 2 is shown in Figure 18. The reducing constants q(1,2;1), q(2,3;2), q(1,2;3) and q(4,5;4) are equal to 3,5,3, and 1 respectively so that the label of the first node of the tree is 12.

In Step 3 $\theta(i_o, j_o; k)$ is found to be $\theta(4,5;4)$ which equals 3. In the modified algorithm the Y node for the next branch is chosen to be (4,5 or 5,4;4) and $\overline{Y}$ is the complement ($\overline{4,5}$ and $\overline{5,4}$;4) which is labeled by $W(X) + \theta(4,5;4) = 15$ in Step 4.

Since no legs have been committed yet in this first iteration, Step 5C is executed immediately following the first substep of Step 5A. $M'_4$ is deleted. All the elements in $M'_3$ except columns 4 and 5 are deleted. Columns and rows 4 and 5 are deleted in $M'_1$ and $M'_2$. The set of matrices formed by restricting the set of $M'_k$ are relabeled as the set of $M_k$ which are to be operated upon in Step 6 and which are shown in Figure 19.

In Step 6 $q(3,4;3) = 4$ is the only non-zero reducing constant so that Y is labeled by 16. Since only one leg of the route has been committed, Step 8 follows Step 7.

In Step 8 ($\overline{4,5}$ and $\overline{5,4}$;4) is selected as the next node from which to branch.

In Step 9 $g = 0$ since no legs have been committed in the subset being partitioned by node X. $M_k$ is set equal to $R_k$ for all k. Substep 3 requires no action since there are no committed arcs and legs. $r^4_{45}$ is set equal to infinity in order to block arc (4,5). The set of $M_k$ is reduced by substep 5 and X is labeled by the sum of the reducing constants which equals 15. Step 3 which is executed next begins the second iteration.

$M_1'$

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 0 | 8 | 11 | 3 |
| 2 | | 4 | 6 | 12 |
| 3 | | | 26 | 1 |
| 4 | | | | 2 |

$M_2'$

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 1 | 6 | 7 | 2 |
| 2 | | 0 | 5 | 8 |
| 3 | | | 10 | 9 |
| 4 | | | | 13 |

$M_3'$

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 0 | 11 | 6 | 26 |
| 2 | | 21 | 5 | 12 |
| 3 | | | 4 | 14 |
| 4 | | | | 2 |

$M_4'$

| | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 16 | 10 | 8 | 21 |
| 2 | | 15 | 9 | 18 |
| 3 | | | 3 | 20 |
| 4 | | | | 0 |

Reduced cost matrices at the end of Step 2.

Figure 18

$M_1$

| | 2 | 3 |
|---|---|---|
| 1 | 0 | 8 |
| 2 | | 4 |

$M_2$

| | 2 | 3 |
|---|---|---|
| 1 | 1 | 6 |
| 2 | | 0 |

$M_3$

| | 4 | 5 |
|---|---|---|
| 1 | 6 | 26 |
| 2 | 5 | 12 |
| 3 | 4 | 14 |

Cost matrices at end of Step 5C of first iteration.

Figure 19

$M_1'$

| | 2 | 3 |
|---|---|---|
| 1 | 0 | 8 |
| 2 | | 4 |

$M_2'$

| | 2 | 3 |
|---|---|---|
| 1 | 1 | 6 |
| 2 | | 0 |

$M_3'$

| | 4 | 5 |
|---|---|---|
| 1 | 2 | 22 |
| 2 | 1 | 8 |
| 3 | 0 | 10 |

Reduced cost matrices at start of third iteration.

Figure 20

Second Iteration

At the end of Step 6, Y = (3,4 or 4,3;4) has a label of 21 and $\overline{Y}$ as a result of Step 4 has a label of 20. In Step 8 (4,5 or 5,4;4) has the smallest label and is selected as the branch point for the next iteration.

In Step 9 $g = r^4_{45} = 1$ since there is only the one committed leg. $M'_k$ is set equal to $R_k$ for all k. Step 5C is carried out for the single committed but undirected leg 4 with arc (4,5). No paths need be blocked in in substep 4. After Step 9 is completed the set of cost matrices formed for the next iteration is given in Figure 20. This is the same set of matrices which would have been formed by reducing the set of matrices after Step 5C in the first iteration which are shown in Figure 19.

Third Iteration

$\theta(i_1, j_o; k_o) = \theta(1,2;1) = 4$ in Step 3, so Y is (1,2 or 2,1;1). As a result of Step 4 $\overline{Y}$ is labeled by 20.

Since leg 2 has not yet been committed, the direction of the route is still not determined, and Step 5C follows Step 5A. $M'_1$ is deleted. Row 3 is deleted from $M'_2$. After being redesignated as $M_k$, the new $M_2$ and $M_3$ are shown in Figure 21.

In Step 6 all reducing constants are zero causing Y to be labeled by 16. In Step 8 Y becomes the chosen branch point for the next iteration. Step 9 is not required in the present iteration.

Fourth Iteration

$\theta(i_o, j_o; k_o) = \theta(3,4;3)$ in Step 3. Y is then (3,4 or 4,3;3). $\overline{Y}$ is labeled by 26 in Step 4. Only the direction (3,4) is compatible

$M_2$

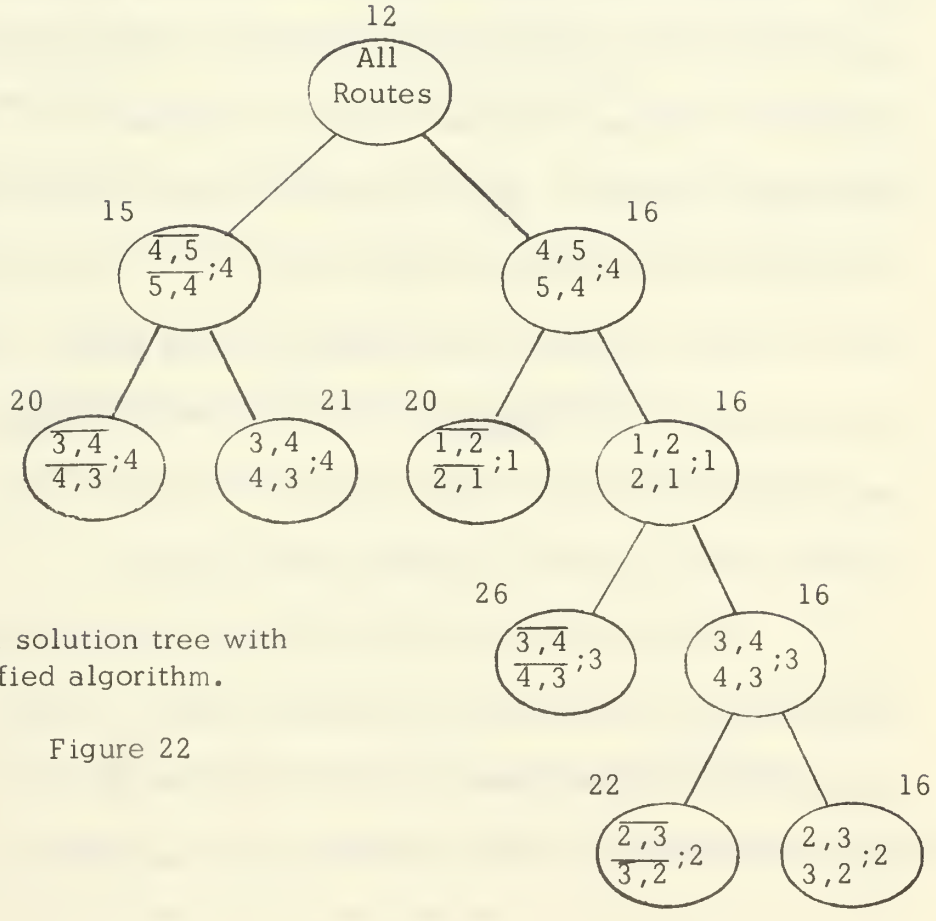
| | 3 |
|---|---|
| 1 | 6 |
| 2 | 0 |

$M_3$

| | 4 | 5 |
|---|---|---|
| 3 | 0 | 10 |

End of Step 5B for Third Iteration.

Figure 21




Final solution tree with modified algorithm.

Figure 22

with the arcs specified for leg 4. Leg $(k_o + 1)$ is leg 4, which is committed and undirected, so that substep 3.a is applicable. Therefore, leg 3 becomes directed (3,4) and leg 4 becomes directed (4,5). Step 5 follows Step 5B. At the end of Step 6 only two elements in the single matrix $M_2$ remain, $m^2_{12} = 6$ and $m^2_{23} = 0$. Consequently the reducing constant is zero and Y is labeled by 16. The next X for branching in Step 8 is again the Y at the end of Step 6.

Final Iteration

The single zero element leads to $(i_o, j_o; k_o)$ equal to (2,3;2) in Step 3. $\overline{Y}$ is labeled with 16 + 6 = 22 in Step 4. For Step 5A leg $(k_o + 1)$ is leg 3 which is already directed so that substep 4.a is applicable and leg 2 becomes directed (2,3). Since leg 1 is undirected substep 4 is repeated, and leg 1 becomes directed (1,2). In Step 5 the last matrix is deleted. Since all legs have been committed, in Step 6 Y retains the same label as X which is 16. (n - 1) legs of the route have now been assigned so that Step 10 follows Step 7. In Step 10 $Z_o$ is set equal to 16. In Step 8 $Z_o$ is found to be less than or equal to the smallest label on any terminal node. Therefore, the optimal route has been isolated and the algorithm is terminated.

The final solution tree is given in Figure 22. Only one connected path is contained in the nodes listed in the tree. The set of directed legs form the optimal route (1,2), (2,3), (3,4), (4,5). A comparison of Figures 16 and 22 shows that three branchings in the leftmost

portion of the tree are eliminated by the use of the modified algorithm. For large problems the modified algorithm is expected to produce significant savings in storage and computer run time.

## F. ADDITIONAL APPLICATIONS

Numerous applications may be constructed which specify arrival times, departure times, lay-overs, and other aspects of a routing problem. The following examples show that certain kinds of constraints can be quite easily entered into the algorithm.

### 1. A Constraint Specifying an Arrival Date for a Given City

Suppose travel times vary because of transportation schedules and the objective is to minimize travel time with the additional constraint that the route must arrive at city m on the $k^{th}$ day of travel over the route. The constraint is easily entered by restricting $R_k$ to column k and $R_{k+1}$ to row m, since arrival at city m must occur on leg k and departure from city m must occur on leg (k + 1).

### 2. A Constraint Specifying an Arc of the Route

Suppose the objective is a minimum cost route with the constraint that one leg will connect city p to city q. The constraint is entered by blocking all arcs from city p to cities other than q and blocking all arcs to q from other cities than p in all of the set of $M_k$.

### 3. A Minimum Weight-Distance Shipping Route Problem

A domestic firm has ordered 8 units of heavy industrial equipment from a foreign manufacturer and desires to minimize the costs of

transporting these units to the plant sites that require them. The manufacturer will deliver all the units to a single point of entry. The firm may choose any one of the plant sites as the point of entry without any additional charges by the manufacturer. The firm must then use a special van to transport and install each of the units of equipment in each of the several plant sites.

The cost of transportation of the units is proportional to the weight-distance hauled. In addition, a minimum cost per unit per day on board is incurred when the special van is not moved a minimum distance in a day. Each unit of equipment weighs the same. The number of units for each plant site is given below.

| Number of units | Plant Number |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 1 | 3 |
| 2 | 4 |
| 1 | 5 |
| 1 | 6 |

The distance between sites is symmetric, so the transportation costs per unit for haulage between sites is symmetric and is given in the triangular matrix in Figure 23.

A unit can be installed on the same day that it arrives at a plant site. But no more than one unit can be installed on a single day, so that charges for a day of storage result for each additional unit installed at the same site.

| | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 1 | 31 | 66 | 75 | 71 | 93 |
| 2 | | 46 | 45 | 38 | 48 |
| 3 | | | 11 | 51 | 54 |
| 4 | | | | 49 | 44 |
| 5 | | | | | 23 |

Cost per unit hauled between plant sites.

Figure 23

| | 2A | 2B | 3 | 4A | 4B | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 1 | 31 | 31 | 66 | 75 | 75 | 71 | 93 |
| 2A | | 8 | 46 | 45 | 45 | 38 | 48 |
| 2B | | | 46 | 45 | 45 | 38 | 48 |
| 3 | | | | 11 | 11 | 51 | 54 |
| 4A | | | | | 8 | 49 | 44 |
| 4B | | | | | | 49 | 44 |
| 5 | | | | | | | 23 |

$(a_{ij})$ Matrix

Figure 24

The first unit is installed at the time of arrival of the entire shipment at the first plant site. Thereafter the costs are based on the number of units hauled or stored until the special van is empty.

Site 2 and site 4 are made into two sites each so that the storage cost of 10 per unit per day may be entered into the cost matrix as shown in Figure 24. The problem is now in the form amenable to solution by the routing algorithm. A unit of equipment is installed on each leg of the route, and the cost for the leg is the product of one element from the cost-per-unit matrix times the number of units left at the beginning of that leg. In the notation of Section III. A. $(c_1, c_2, \ldots, c_7) = (7, 6, \ldots, 1)$, each $c_k$ is the number of units yet to be installed at the beginning of the $k^{th}$ leg. $(a_{ij})$ is the cost matrix of Figure 24. $R_k = c_k(a_{ij})$ for $k = 1, 2, \ldots, 7$. Since each of the $R_k$ is symmetric, the algorithm as modified in Section III. E. may be used to find the sequence of deliveries of the equipment which minimizes the cost to the firm.

4. Problems Including Both Cost and Profit

Although only non-negative elements have been used in the cost matrices of the examples, negative costs are permitted; and, in fact, with the use of negative elements, negative labels may occur in the tree constructed by the algorithm. Consequently, net costs after subtracting profits for a given day's travel may be entered in the cost matrices. The algorithm is then used to produce the optimal net cost route which will have value less than zero if there is an overall net profit.

## G. SPECIAL CASES SOLVED BETTER BY OTHER METHODS

### 1. Identical Cost Matrices for Each Leg

Suppose the optimum route is desired where $R_i = R_j$ for all i,j. This problem may be more efficiently solved by converting it to a traveling salesman problem by injecting one more city, say city (n + 1), and specifying that $r_{n+1,j} = r_{i,n+1} = 0$ for $i,j = 1,2,\ldots,n$. The recommendations of Ref. 1 may then be followed in finding an efficient way to solve the problem. The optimal traveling salesman route will contain the optimal route through the original n cities. The beginning and end of the optimal route desired are the cities which connect with the artificial city that was injected to obtain a solution.

### 2. Cost Rate Constants Form a Monotonic Sequence

If the optimal route is to be found from a set of $(r_{ij}^k)$ matrices where $r_{ij}^k = c_k a_{ij}$ and if the set of $c_k$ and the matrix $(a_{ij})$ have special properties, the optimal route may be found very simply without recourse to this algorithm.

If the set of $c_k$ form a monotonically decreasing sequence for $k = 1,2,\ldots,(n-1)$, and if the shortest route using the distances in $(a_{ij})$ is found to be a monotonically increasing sequence of arc lengths, the least cost route is the same as the shortest route of $(a_{ij})$. Furthermore, the cost of the optimal route is the sum of the pairwise products of the elements taken in order from each of the two sequences. The same result is true if the directions of monotonicity were reversed for both sequences.

A theorem with respect to the minimum sum of pairwise products of the elements of an ascending and a descending sequence is presented in Ref. 5.

## IV. DISCUSSION

The sequence dependent route through n cities is closely akin to the traveling salesman problem. In fact if an additional leg were added to close the circuit back to the start of the route, the problem would be that of the traveling salesman with route costs changing after each arc is traversed. Consequently, the literature on the traveling salesman problem - in particular Little, et al [Ref. 1], the survey of the traveling salesman problem by Bellmore and Nemhauser [Ref. 2], and the survey of the branch-and-bound method by Lawler and Wood [Ref. 6] - contain much relevant comment.

### A. BRANCH-AND-BOUND COMPARED WITH DYNAMIC PROGRAMMING

The number of computations, the storage space, and solution time required for the solution of the optimal route problem are exactly determined by the number of elements in the cost matrices when dynamic programming is the solution method. These three factors are quite variable for the branch-and-bound method. All three depend on the number of nodes which must be generated in the tree to solve the problem. In many cases, such as the example in Section III. B. branch-and-bound is superior to dynamic programming in all three factors. Comparison for the traveling salesman problem in Ref. 2 shows that branch-and-bound provides lower average solution times with a much larger variance than similar problems solved by dynamic programming. For practical

purposes dynamic programming is restricted to the solution of routing problems with fewer than 15 nodes on the route. The storage requirements for the values of the recursive functions in dynamic programming for a 15 node problem is 24,024 [Ref. 2]. For larger problems branch-and-bound methods prove superior by reducing the storage requirement.

## B. STRATEGIES OF BRANCHING

The strategy used for the selection of the next branch point has a direct effect on the storage requirements for the branch-and-bound algorithm to solve the routing problem. The policy used in the flow chart is to branch from the lowest bound. This policy has the advantage of minimizing total computation, in the sense that any branching performed is also that which must be performed under any alternate policy. However, under a "branch from the lowest bound" policy, no terminal nodes are discarded and the volume of storage may become excessive. The alternate strategy is to branch always from the newest active node and to discard nodes from storage that are out of the competition for the optimal route. Very few nodes are then kept in storage --- on the order of a few n [Ref. 1]. For this second strategy in the routing algorithm, branching would always be to the right until a complete route was enumerated or the bound exceeded the cost of a complete route already found. Then the process would work backwards up the branch until a left hand node with lower bound less than the cost of the completed route was found. The nodes that were passed by could be discarded from

storage and branching would be continuously to the right again from the selected node. This procedure saves storage but increases the number of computations and the solution time because many nodes are calculated unnecessarily when compared with the first strategy.

## C. SUBOPTIMIZATION

A very useful feature of the branch-and-bound routing algorithm is that it allows accepting the solution and terminating the computation when a satisfactory suboptimization solution has been found. For instance, if the objective is to find a route that meets a budget constraint, the branch to the right strategy may be used, and the computation ends as soon as a single route with cost less than the budget constraint has been found; or using the branch from the lowest bound policy, computation ends when it is determined that no route less than the budget constraint exists. In addition Ref. 6 gives branch-and-bound search procedures that are applicable to the routing algorithm for finding suboptimal solutions that are within a given percentage of the optimal and also for finding the best suboptimal solution under a solution time constraint.

# V. CONCLUSIONS

The branch-and-bound algorithm can successfully find the optimal route for a variety of sequence dependent routing problems, including those with symmetric cost matrices, when the matrices of all possible costs for each leg of the route is known. On the basis of experience with the traveling salesman problem, the branch-and-bound routing algorithm is expected to be competitive with dynamic programming for routes involving fewer than 15 nodes and superior for larger problems. Under the branch-and-bound algorithm, storage requirements may be traded for increases in solution time and the number of computations. This trade-off is accomplished by the selection of the branching strategy used in the algorithm. Experience from digital computer runs is necessary to provide more quantitative recommendations for branching procedures. Finally, only the branch-and-bound algorithm provides a rapid suboptimal solution when these are acceptable.

## BIBLIOGRAPHY


1. Little, J.D.C., and others, "An Algorithm for the Traveling Salesman Problem," Operations Research, v. 11, p. 972-989, 1963.

2. Bellmore, M., and Nemhauser, G.L., "The Traveling Salesman Problem: A Survey," Operations Research, v. 16, p. 538-558, 1968.

3. Golomb, S.W., and Baumert, L. D., "Backtrack Programming," Journal of the Association for Computing Machinery, v. 12, n. 4, p. 516-524, October 1965.

4. Ford, L.R., and Fulkerson, D.R., Flows in Networks, p. 2-3, Princeton University Press, 1962.

5. Gavett, J.W., and Plyter, N.V., "Optimal Assignments of Facilities to Locations by Branch and Bound," Operations Research, v. 14, p. 210-2, 1966.

6. Lawler, E.L., and Wood, D.E., "Branch-and-Bound Methods: A Survey," Operations Research, v. 14, p. 699-719, 1966.

7. Bellman, R.E., "Dynamic Programming Treatment of the Traveling Salesman Problem," Journal of the Association for Computing Machinery, v. 9, p. 61-63, 1962.

8. Cooke, K.L., and Halsey, E., "The Shortest Route through a Network with Time-Dependent Internodal Transit Times," Journal of Mathematical Analysis and Applications, v. 14, p. 493-498, 1966.

9. Dantzig, G.B., All Shortest Routes in a Graph, Operations Research House, Stanford University Technical Report 66-3, November 1966.

10. Dreyfus, S.E., "An Appraisal of Some Shortest-Path Algorithms," Operations Research, v. 17, p. 395-412, May-June 1969.

11. Floyd, R.W., "Algorithm 97, Shortest Path," Communications of the Association for Computing Machinery, v. 5, p. 345, 1962.

12. Hayes, R.L., The Delivery Problem, Carnegie Institute of Technology, Management Science Research Report No. 106, July 1967.

13. Held, M., and Karp, R.M., "A Dynamic Programming Approach to Sequencing Problems," Journal of the Society of Industrial and Applied Mathematics, v. 10, p. 196-210, March 1962.

14. Hu, T.C., "Revised Matrix Algorithms for Shortest Paths," Journal of the Society of Industrial and Applied Mathematics, v. 15, p. 207-218, 1967.

INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 20 |
| 2. | Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Director, Systems Analysis Division (Op-96)<br>Office of the Chief of Naval Operations<br>Navy Department<br>Washington, D. C. 20350 | 1 |
| 4. | Department of Operations Analysis, Code 55<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 5. | Associate Professor Rex H. Schudde<br>Department of Operations Analysis<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | Assistant Professor G. T. Howard<br>Department of Operations Analysis<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 7. | Special Projects Office<br>Department of the Navy (Sp-114)<br>Washington, D. C. 20360<br>Attn: CDR W. Thompson, USN | 1 |
| 8. | LCDR Michael J. DeHaemer, USN<br>USS Whale (SSN-638)<br>FPO New York, New York 09501 | 1 |

Security Classification

## DOCUMENT CONTROL DATA - R & D

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

| 1 ORIGINATING ACTIVITY (Corporate author) | 2a. REPORT SECURITY CLASSIFICATION |
|---|---|
| Naval Postgraduate School<br>Monterey, California 93940 | Unclassified |
| | 2b. GROUP |

3 REPORT TITLE

A Branch-and-Bound Algorithm for the Solution of Sequence Dependent Routing Problems

4 DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; April 1970

5 AUTHOR(S) (First name, middle initial, last name)

Michael Joseph DeHaemer

| 6 REPORT DATE | 7a. TOTAL NO. OF PAGES | 7b. NO. OF REFS |
|---|---|---|
| April 1970 | 57 | 14 |

| 8a. CONTRACT OR GRANT NO | 9a. ORIGINATOR'S REPORT NUMBER(S) |
|---|---|
| b. PROJECT NO | |
| c. | 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) |
| d. | |

10 DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

| 11 SUPPLEMENTARY NOTES | 12. SPONSORING MILITARY ACTIVITY |
|---|---|
| | Naval Postgraduate School<br>Monterey, California 93940 |

13 ABSTRACT

A branch-and-bound algorithm, which finds the optimal route through n nodes when a different cost matrix occurs after each arc in the sequence is traversed, is presented. The route may begin at any node and must pass through each of the n nodes exactly once. The objective is to minimize total cost in traversing (n-1) arcs of the route. The cost of traversing each arc is $r_{ij}^k$, which is a function of the distance between nodes i and j and a function of the $k^{th}$ position in the sequence of arcs forming the route.

The algorithm is presented in step-by-step detail and illustrated by flow chart and examples. A modification for symmetric ($r_{ij}^k$) improves the efficiency of the algorithm.

A trade-off between computation time and storage requirements may be accomplished by alternate branching policies. Suboptimal solutions may be obtained with reduced computation.

DD FORM 1 NOV 65 1473 (PAGE 1)



S/N 0101-807-6811

Security Classification

A-31408

| 14 KEY WORDS | LINK A | | LINK B | | LINK C | |
|---|---|---|---|---|---|---|
| | ROLE | WT | ROLE | WT | ROLE | WT |
| Routing | | | | | | |
| Sequencing | | | | | | |
| Traveling Salesman | | | | | | |
| Branch-and-Bound | | | | | | |
| Algorithm | | | | | | |
| Shortest Route | | | | | | |

Thesis
D246
c.1

118474

DeHaemer
A branch-and-bound algorithm for the solution sequence dependent routing problem.